# Fluid Simulation

**Jungdam Won**

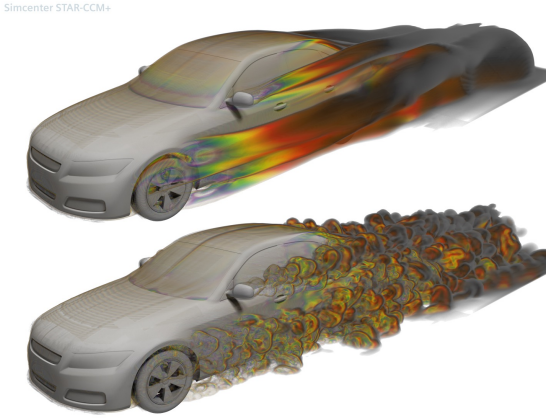
Computer Science & Engineering

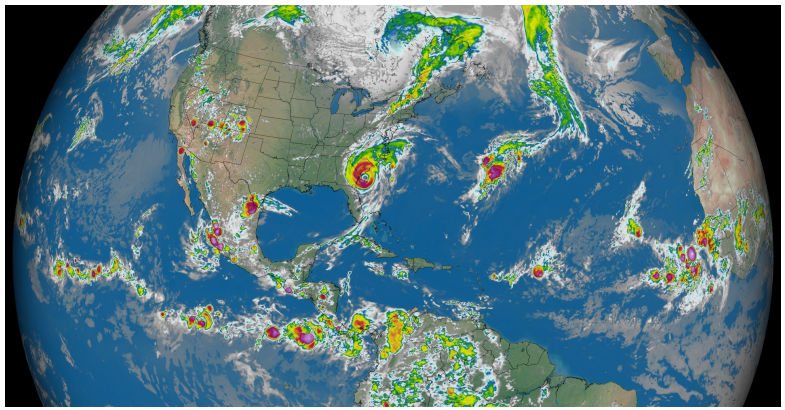
Seoul National Univ.

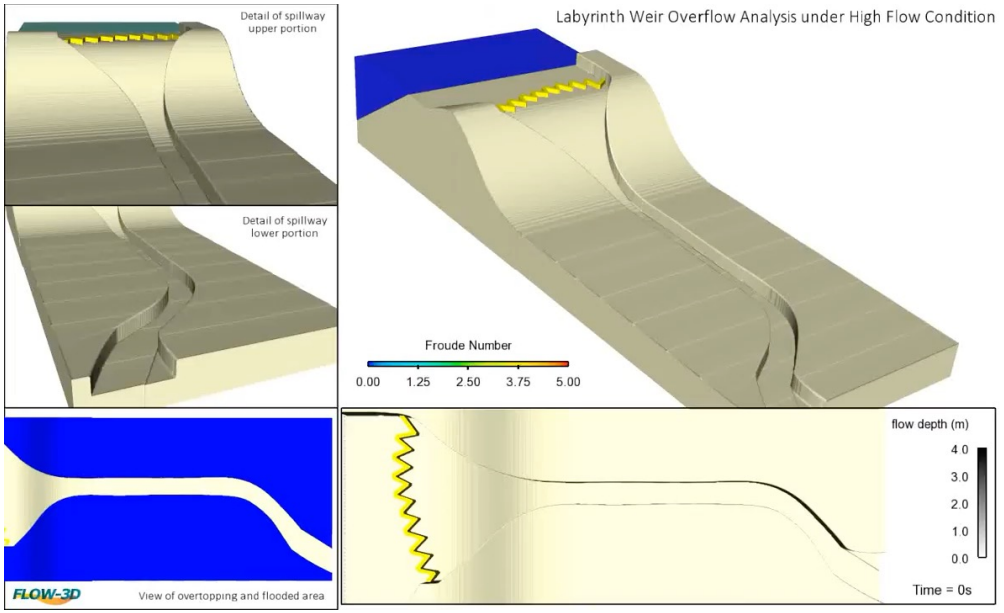
This material was created based on the slides and lecture notes of *Fluid Simulation for Computer Animation* (SIGGRAPH 2007 course) by Robert Bridson Matthias and Müller-Fischer, and a course noted made by Prof. Stelian Coros

# Applications: Engineering

- It has been widely used in many enginnering problems





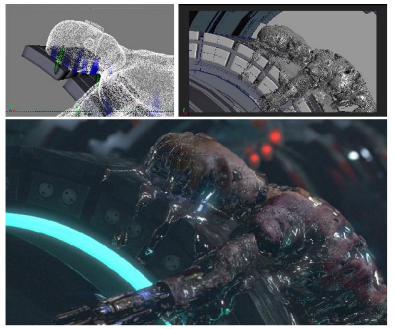

[Computational fluid dynamics – Wikipedia]
[NASA@SC19: The Future of Global Numerical Weather Prediction with GEOS]
[Dams and Spillways - FLOW-3D HYDRO | CFD Modeling Solution (flow3d.com)]

# Applications: Film Making

- It has been widely used for visual effects in film industry

[Antz, 1998]


[Shrek, 2001]



[Terminator 3, 2003]

[Pirates of the Caribbean 3, 2007]

[The Day After Tomorrow, 2004]

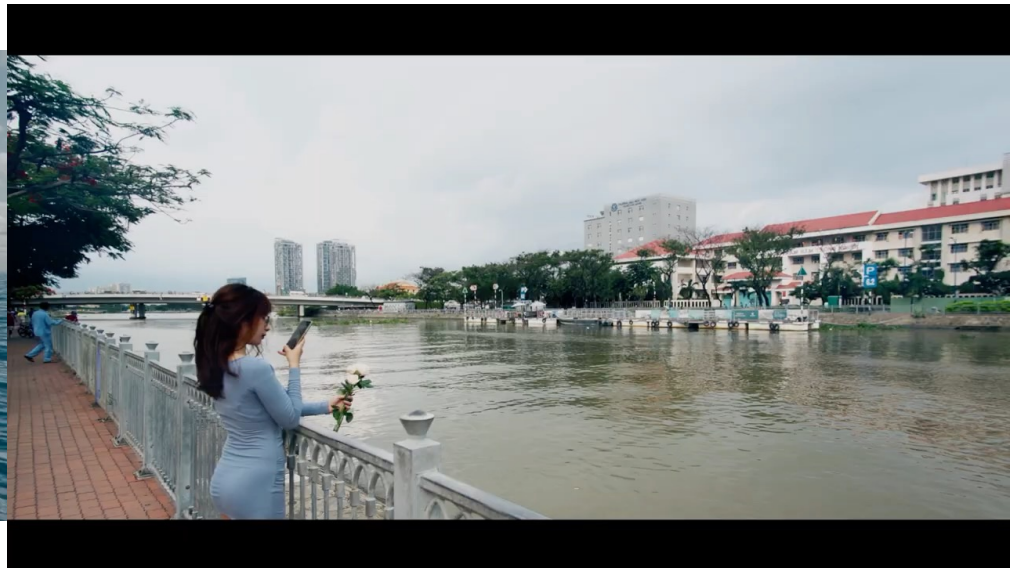
And many others…

# Applications: Film Making

- In CG, creating a ***visually pleasing*** dynamic effects of fluids in a ***reasonable amount of time*** is far more important than how close the simulated result is to the real world

- In addition to the requirements above, ***controllability*** is one of the key features required
  - For example, a director might want a part of water to be separated from the sea at the right ***time*** and in the right ***place*** when the actors are performing something important

# Videos: Film Making



[Water FX in Houdini | Pro Online Course (Extended BTS Trailer) – YouTube]
[VFX Scene Breakdown - Houdini FLUID Simulation Sci-fi SHIP - YouTube]

# Applications: Games

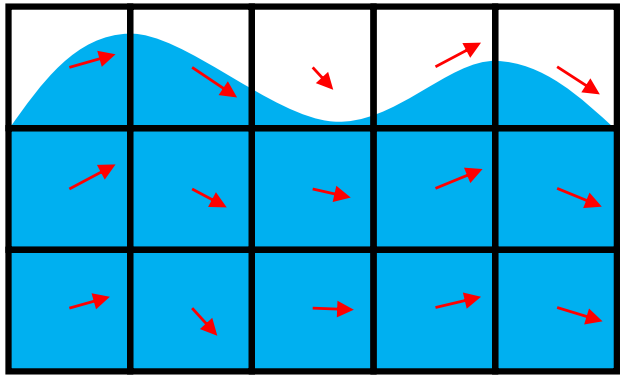
- In old days, fluid simulation had not been adopted in many computer games due to tight computational budget
  - In many games, fluids have still been animated procedually by using sinusoidal functions

- Recently, there have been several attempts to develop realtime fluid simulation, where ***Interactivity*** is the key feature because users can intervene what happens in the scene when compared to making films
  - Model reduction (e.g. 2.5D)

https://youtu.be/_3eyPUyqluc?t=413

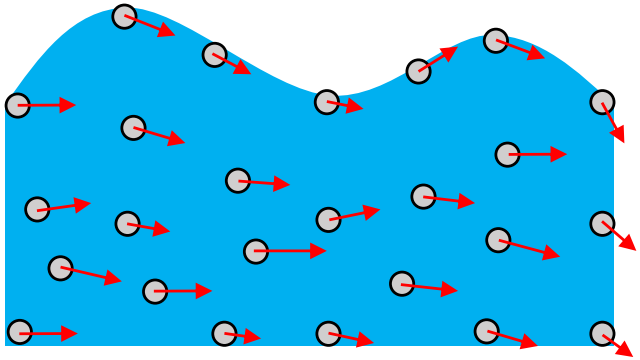
# Eulerian vs. Lagrangian

- Eulerian viewpoint



- We measure fluid quantities (e.g. velocity, pressure) in fixed locations

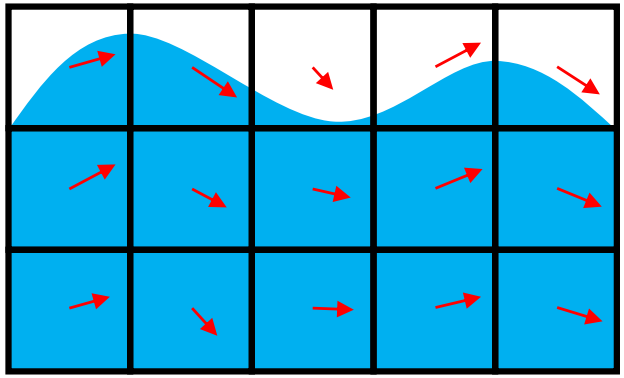
- Lagrangian viewpoint



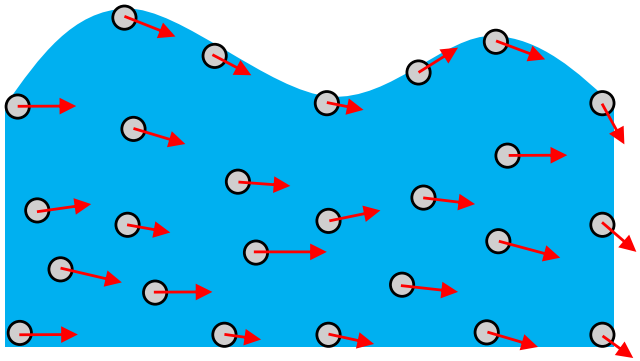
- Particles represents fluid, where we measure fluid quantities each particle has

# Eulerian vs. Lagrangian

- Eulerian viewpoint



- Lagrangian viewpoint



When doing a weather report

You are ***stuck*** on the ground, measuring the pressure, temperature, humidity, and etc. of the air that's flowing past

You are in the balloon floating along with the wind, measuring the pressure, temperature, humidity, and etc. of the air that's flowing ***alongside*** you

# Total/Material Derivative

- Let's assume we want to measure a physical quantity (e.g. temperature, pressure) at time $t$ and in the position $(x, y, z)$

$$q(t, x, y, z)$$

- Then, its time derivative is:

$$\frac{dq(t,x,y,z)}{dt} = \frac{\partial q}{\partial t} + \frac{\partial q}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial q}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial q}{\partial z}\frac{\partial z}{\partial t}$$

The change of quantity itself
(e.g. the air temperature goes up due to heat)

The change of quantity occurred due to its movement, which is only non-zeros in Eulerian viewpoint
(e.g. the air temperature at the observation location can go up when hot air is blowing to the location even if no temperature change occurs in the entire system)

# Reminder: Vector Calculus

- ***Gradient*** of scalar-valued function:

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right)$$

- As an operator:

$$\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)$$

# Reminder: Vector Calculus

- ***Divergence*** of vector-valued function:

$$\nabla \cdot \mathbf{u} = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right) \cdot (u, v, w) = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}$$

- As an operator:

$$\nabla \cdot = \frac{\partial}{\partial x} + \frac{\partial}{\partial y} + \frac{\partial}{\partial z}$$

# Reminder: Vector Calculus

- ***Curl*** of vector-valued function:

$$\nabla\times\mathbf{u} = \left(\frac{\partial}{\partial x},\frac{\partial}{\partial y},\frac{\partial}{\partial z}\right)\times(u,v,w) = \left(\frac{\partial w}{\partial y}-\frac{\partial v}{\partial z},\frac{\partial u}{\partial z}-\frac{\partial w}{\partial x},\frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}\right)$$

- As an operator:

$$\nabla\times = \left(\frac{\partial}{\partial x},\frac{\partial}{\partial y},\frac{\partial}{\partial z}\right)\times$$

# Reminder: Vector Calculus

- ***Laplacian*** of vector-valued function:

$$\nabla \cdot \nabla \mathbf{u} = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right) \cdot \left(\frac{\partial \mathbf{u}}{\partial x}, \frac{\partial \mathbf{u}}{\partial y}, \frac{\partial \mathbf{u}}{\partial z}\right) = \frac{\partial^2 \mathbf{u}}{\partial x^2} + \frac{\partial^2 \mathbf{u}}{\partial y^2} + \frac{\partial^2 \mathbf{u}}{\partial z^2}$$

- As an operator:

$$\nabla^2 = \nabla \cdot \nabla = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

# Navier-Stokes Equation

$$\rho \frac{d\mathbf{u}}{dt} = \rho \mathbf{g} - \nabla p + \rho \nu \nabla^2 \mathbf{u}$$

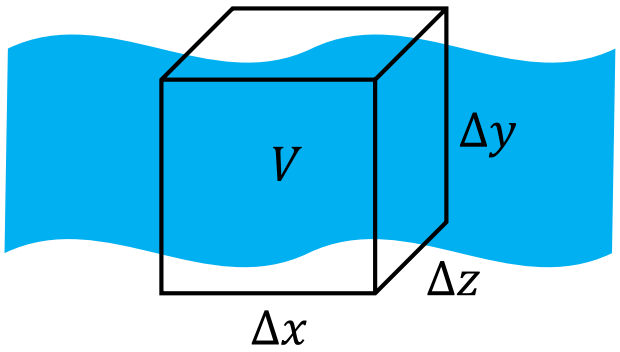
- $\mathbf{u} = (u, v, w)$: Fluid velocity (vector field)
- $p$: Fluid pressure (scalar field)
- $\nu$: Fluid viscosity (constant scalar)
- $\rho$: Fluid density (constant scalar)
- $\mathrm{g} = (0, -9.81, 0)$: Graivitational acceleration (constant vector)

# Navier-Stokes Equation

$$\rho \frac{d\mathbf{u}}{dt} = \rho\mathbf{g} - \nabla p + \rho\nu\nabla^2\mathbf{u}$$

$\mathrm{m}\boldsymbol{a}$ $\qquad\qquad$ $\mathbf{f}_{\mathrm{int}} + \mathbf{f}_{\mathrm{ext}}$

- The equation is just a reformulation of $F = m\boldsymbol{a}$ for fluids when being considered in an infinitesimal control volume $V$



# Change in Velocity

$$\rho \frac{d\mathbf{u}}{dt} = \rho \mathbf{g} - \nabla p + \rho \nu \nabla^2 \mathbf{u}$$

$$\frac{m}{V}\frac{d\mathbf{u}}{dt} = \frac{m}{V}\boldsymbol{a}$$

- The first term, $\frac{d\mathbf{u}}{dt}$, is the ***total derivative*** of fluid velocity with respect to time, which represents how fast the velocity changes in the control volume

$$\frac{d\mathbf{u}}{dt} = \frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{u}}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial \mathbf{u}}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial \mathbf{u}}{\partial z}\frac{\partial z}{\partial t} = \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}$$

# Gravitational Force (Body Force)

$$\rho \frac{d\mathbf{u}}{dt} = \underbrace{\rho \mathbf{g}}_{\frac{m}{V}\mathbf{g}} - \nabla p + \rho \nu \nabla^2 \mathbf{u}$$

- The gravitational force applied to the control volume

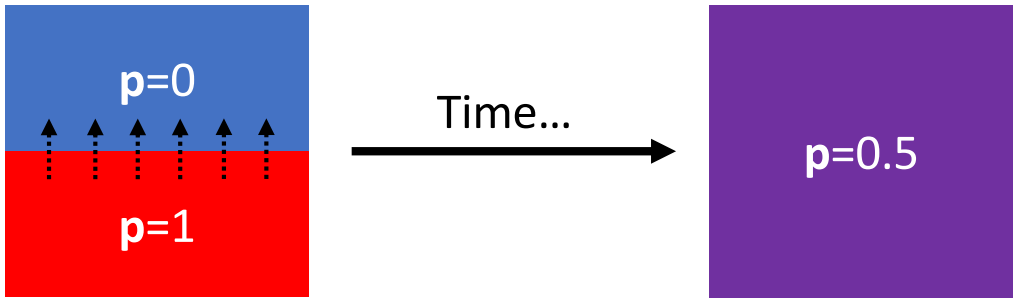
- Other external forces such as winds or user-force can also be added to this term if exists

# Pressure

$$\rho \frac{d\mathbf{u}}{dt} = \rho \mathbf{g} - \nabla p + \rho \nu \nabla^2 \mathbf{u}$$

$$(\frac{f}{A})/h = \frac{f}{V}$$

- Fluid moves (diffuses) from high-pressure areas to low-pressure areas, where it moves in direction of largest change in pressure (i.e. the ***gradient***)



# Diffusion (Viscocity) Term

$$\rho \frac{d\mathbf{u}}{dt} = \rho \mathbf{g} - \nabla p + \rho \nu \nabla^2 \mathbf{u}$$

- Fluids with high viscocity resist change in velocity
  - High-viscosity fluids (e.g. honey) stick together, velocity differences among adjacent elements will eventually converge to zero (i.e. damped) even without external forces
  - Low-viscosity fluids (e.g. gas) flow freely, velocity differences will be kept if no external force is applied

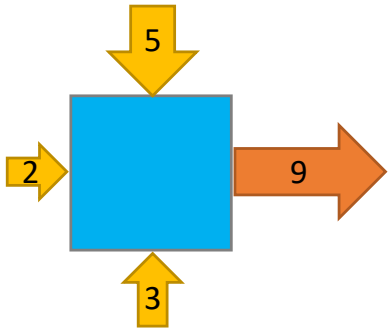
# Summary of NS Equation

$$\rho \frac{d\mathbf{u}}{dt} = \rho \mathbf{g} - \nabla p + \rho \nu \nabla^2 \mathbf{u}$$

Gravity (+ external force) + Pressure + Viscosity
→ Change in velocity

- We will also add another assumption, which is mass conservation (incompressibility) in the control volume
- The amount of ***fluid-in*** equals to the amount of ***fluid-out***

$$\nabla \cdot \mathbf{u} = 0$$

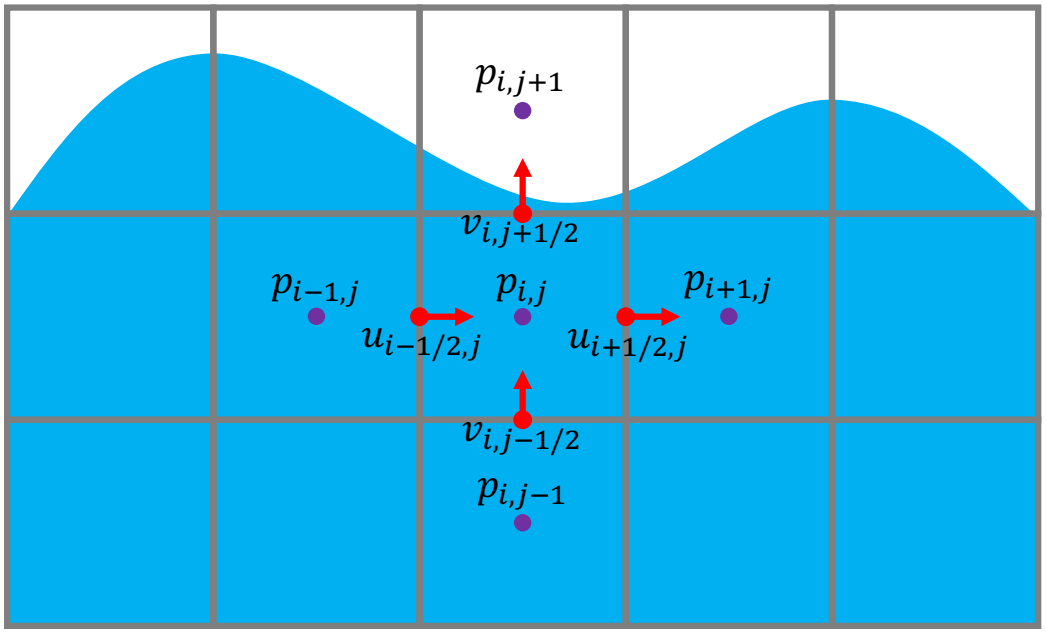


# Algorithm for Fluid Simulation

1. Discretize the simulation space well (spatial Discretization)

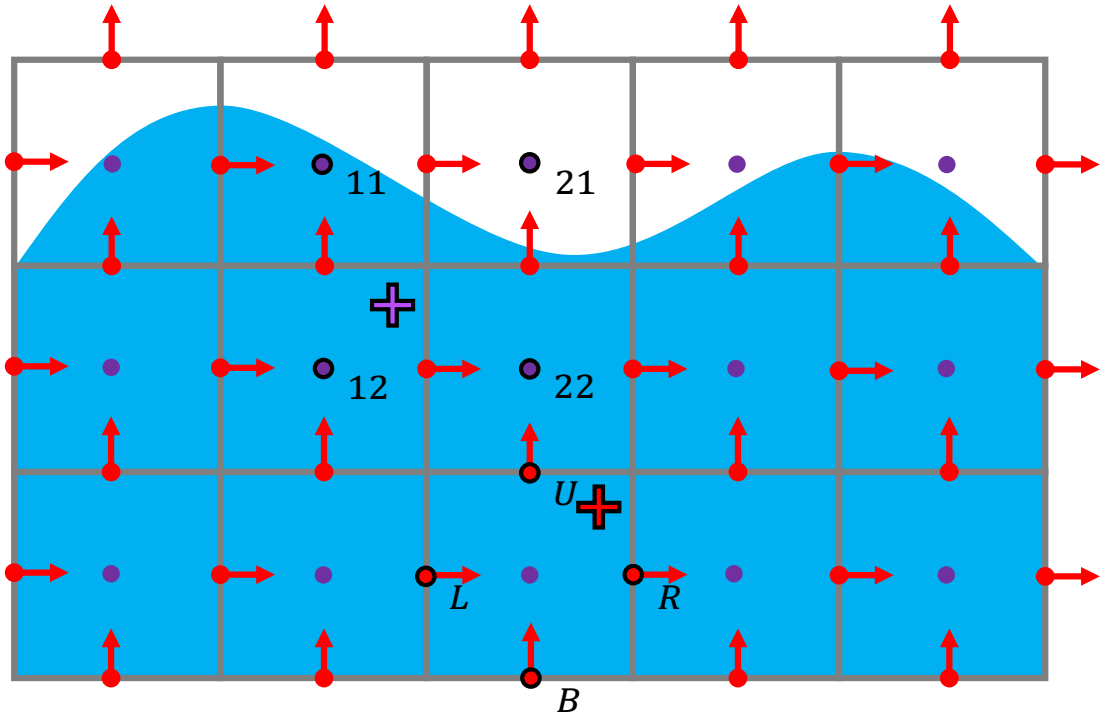
2. Solve NS equation numerically given discretization (numerical integration)

# Spatial Discretization




- Staggered grid (MAC grid)
  - ***Axis-aligned*** cubical cells
  - Scalar quantities (e.g. pressure) are stored at center
  - Velocity compoenents are stored on faces of cells
- Staggering provides an ***unbiased estimate of velocity at center***, and it is more stable and second order accurate
- Divergence, Laplacian, etc. on a grid will be computed via finite difference

# Evaluation at an Arbitary Position



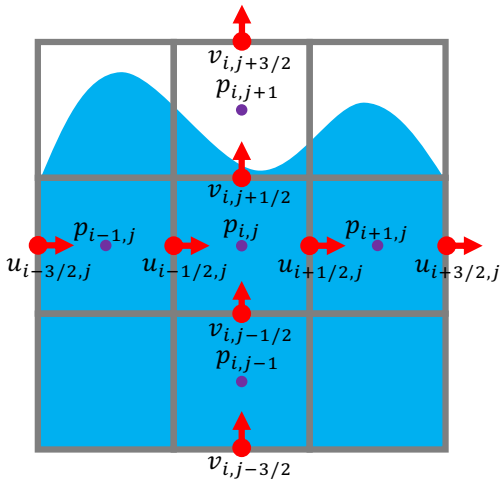

$$f(\oplus) = BilinearInterp(\bullet_{11}, \bullet_{12}, \bullet_{21}, \bullet_{22})$$

$$f_x(\oplus) = LinearInterp(\bullet_L, \bullet_R)$$

$$f_y(\oplus) = LinearInterp(\bullet_U, \bullet_B)$$

# Gradient on a 2D Grid

$$\nabla \mathbf{u}(x,y) = \left(\frac{\partial \mathbf{u}(x,y)}{\partial x}, \frac{\partial \mathbf{u}(x,y)}{\partial y}\right)$$

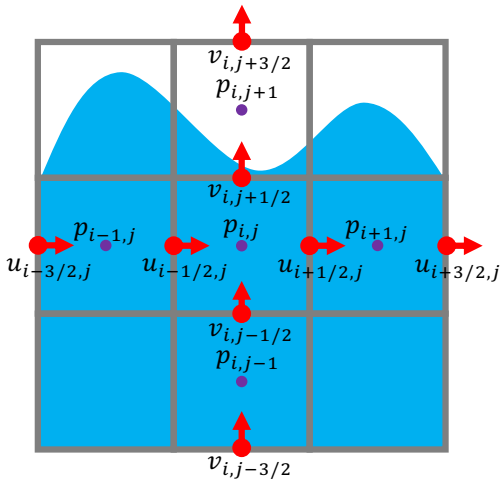



Discretizing on a grid with cell size $h$:

$$\nabla \mathbf{u}(i,j) \approx \left(\frac{\mathbf{u}(i+1/2,j) - \mathbf{u}(i-1/2,j)}{h}, \frac{\mathbf{u}(i,j+1/2) - \mathbf{u}(i,j-1/2)}{h}\right)$$

# Divergence on a 2D Grid

$$\nabla \cdot \mathbf{u}(x,y) = \frac{\partial \mathbf{u}(x,y)}{\partial x} + \frac{\partial \mathbf{u}(x,y)}{\partial y}$$

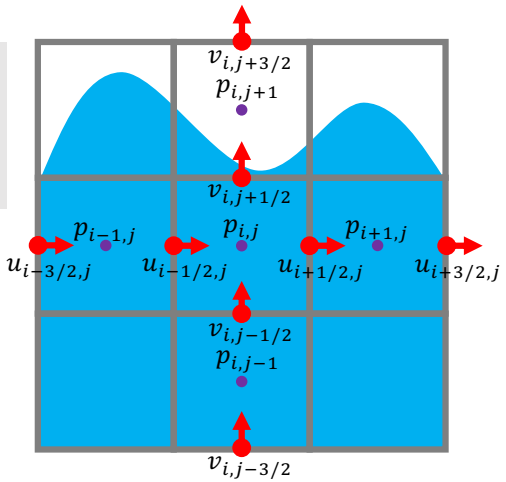



Discretizing on a grid with cell size $h$:

$$\nabla \cdot \mathbf{u}(i,j) \approx \frac{\mathbf{u}(i+1/2,j) + \mathbf{u}(i,j+1/2) - \mathbf{u}(i-1/2,j) - \mathbf{u}(i,j-1/2)}{h}$$

# Laplacian on a 2D Grid

$$\nabla^2 \mathbf{u}(x,y) = \frac{\partial^2 \mathbf{u}(x,y)}{\partial x^2} + \frac{\partial^2 \mathbf{u}(x,y)}{\partial y^2}$$




Discretizing on a grid with cell size $h$:

$$\nabla^2 \mathbf{u}(i,j) \approx$$

$$\frac{\frac{\mathbf{u}(i+1,j)-\mathbf{u}(i,j)}{h} - \frac{\mathbf{u}(i,j)-\mathbf{u}(i-1,j)}{h}}{h} + \frac{\frac{\mathbf{u}(i,j+1)-\mathbf{u}(i,j)}{h} - \frac{\mathbf{u}(i,j)-\mathbf{u}(i,j-1)}{h}}{h}$$

$$= \frac{\mathbf{u}(i+1,j)+\mathbf{u}(i-1,j)+\mathbf{u}(i,j+1)+\mathbf{u}(i,j-1)-4\mathbf{u}(i,j)}{h^2}$$

$$\nabla^2 \mathbf{u} \approx \frac{\mathbf{u}_{\text{right}} + \mathbf{u}_{\text{left}} + \mathbf{u}_{\text{top}} + \mathbf{u}_{\text{bottom}} - 4\mathbf{u}}{h^2}$$

# Algorithm for Fluid Simulation

1. Discretize the simulation space well (spatial Discretization)

2. Solve NS equation numerically given discretization (numerical integration)

# Navier-Stokes Equation

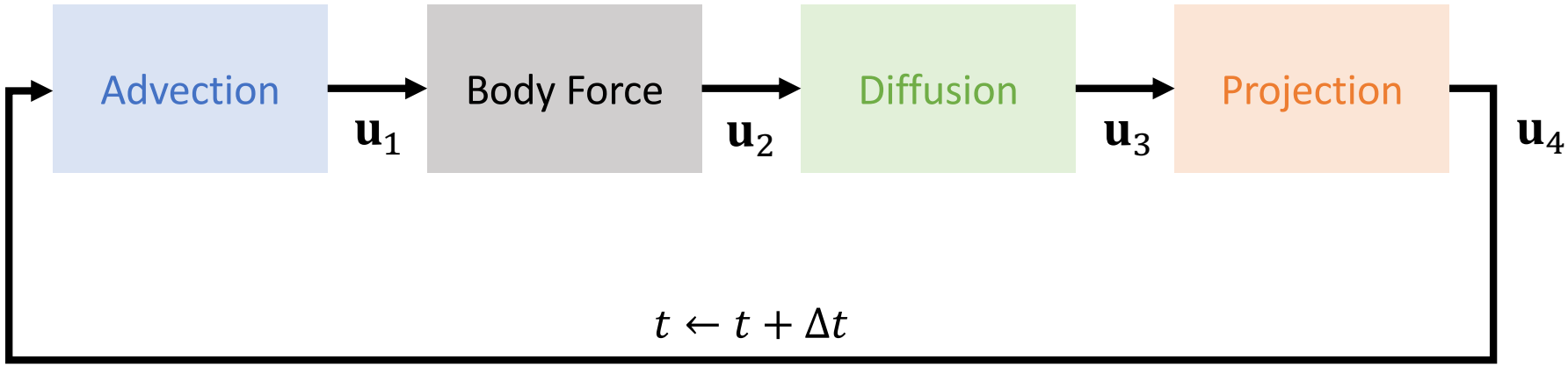
$$\rho \frac{d\mathbf{u}}{dt} = \rho\mathbf{g} - \nabla p + \rho\nu\nabla^2\mathbf{u}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla\mathbf{u} = \mathbf{g} - \frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u}$$

- Solving NS equation all at once is challenging
- Instead, we will solve each term sequentially

# Algorithm (Splitting)

$$\underbrace{\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}}_{(1)} = \underbrace{\mathbf{g}}_{(2)} \underbrace{- \frac{1}{\rho} \nabla p}_{(4)} + \underbrace{\nu \nabla^2 \mathbf{u}}_{(3)}$$



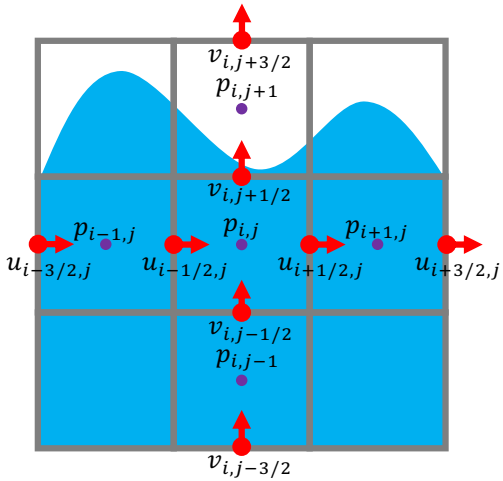
# Advection

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \mathbf{g} - \frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u}$$

(1)

- In this step, the velocity field will be updated as if no external force exists

# Advection: Naïve Update




$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \mathbf{0}$$

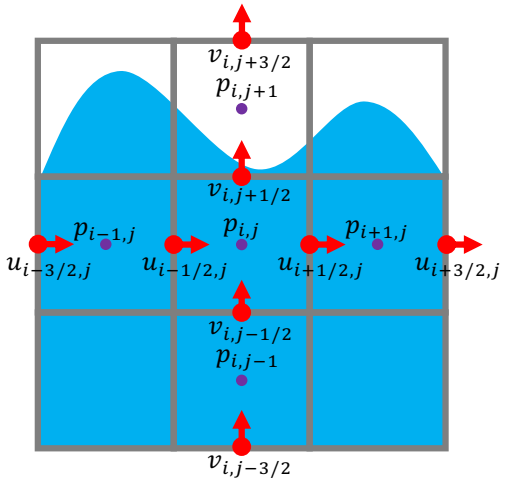
$$\frac{u_{i+1/2,j}^{t+1} - u_{i+1/2,j}^{t}}{\Delta t} + u_{i+1/2,j}^{t} \frac{u_{i+3/2,j}^{t} - u_{i-1/2,j}^{t}}{2h} = 0$$

$$\frac{v_{i,j+1/2}^{t+1} - v_{i,j+1/2}^{t}}{\Delta t} + v_{i,j+1/2}^{t} \frac{v_{i,j+3/2}^{t} - v_{i,j-1/2}^{t}}{2h} = 0$$

$$u_{i+1/2,j}^{t+1} = u_{i+1/2,j}^{t} - \Delta t u_{i+1/2,j}^{t} \frac{u_{i+3/2,j}^{t} - u_{i-1/2,j}^{t}}{2h} = 0$$

$$v_{i,j+1/2}^{t+1} = v_{i,j+1/2}^{t} - \Delta t v_{i,j+1/2}^{t} \frac{v_{i,j+3/2}^{t} - v_{i,j-1/2}^{t}}{2h} = 0$$

# Advection: Naïve Update




$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \mathbf{0}$$

$$\frac{u_{i+1/2,j}^{t+1} - u_{i+1/2,j}^{t}}{\Delta t} + u_{i+1/2,j}^{t} \frac{u_{i+3/2,j}^{t} - u_{i-1/2,j}^{t}}{2h} = 0$$

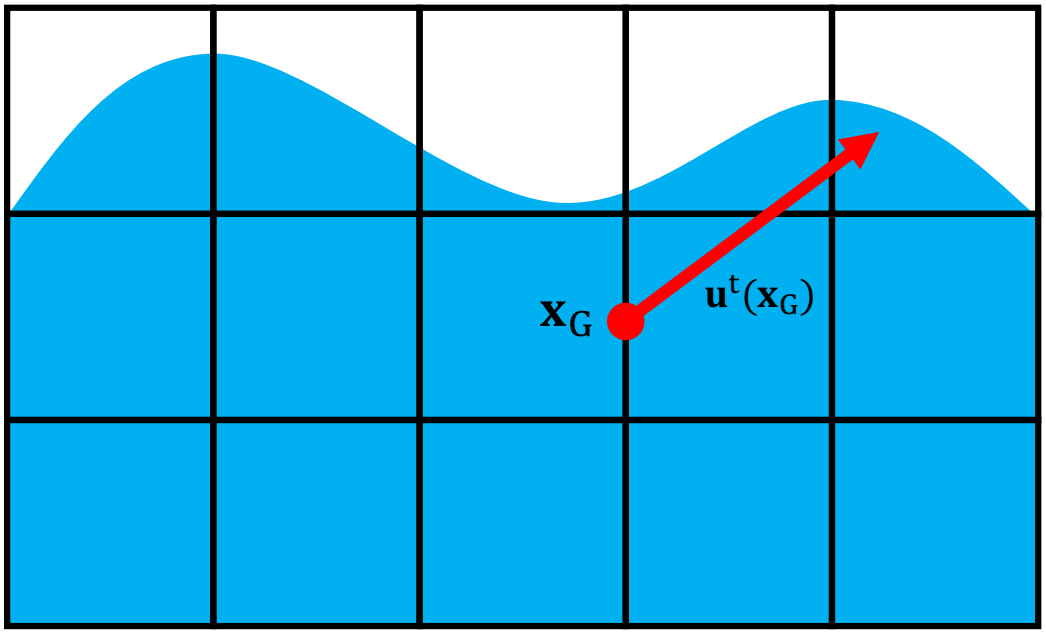
$$\frac{v_{i,j+1/2}^{t+1} - v_{i,j+1/2}^{t}}{\Delta t} + v_{i,j+1/2}^{t} \frac{v_{i,j+3/2}^{t} - v_{i,j-1/2}^{t}}{2h} = 0$$

$$u_{i+1/2,j}^{t+1} = u_{i+1/2,j}^{t} - \Delta t u_{i+1/2,j}^{t} \frac{u_{i+3/2,j}^{t} - u_{i-1/2,j}^{t}}{2h} = 0$$

$$v_{i,j+1/2}^{t+1} = v_{i,j+1/2}^{t} - \Delta t v_{i,j+1/2}^{t} \frac{v_{i,j+3/2}^{t} - v_{i,j-1/2}^{t}}{2h} = 0$$

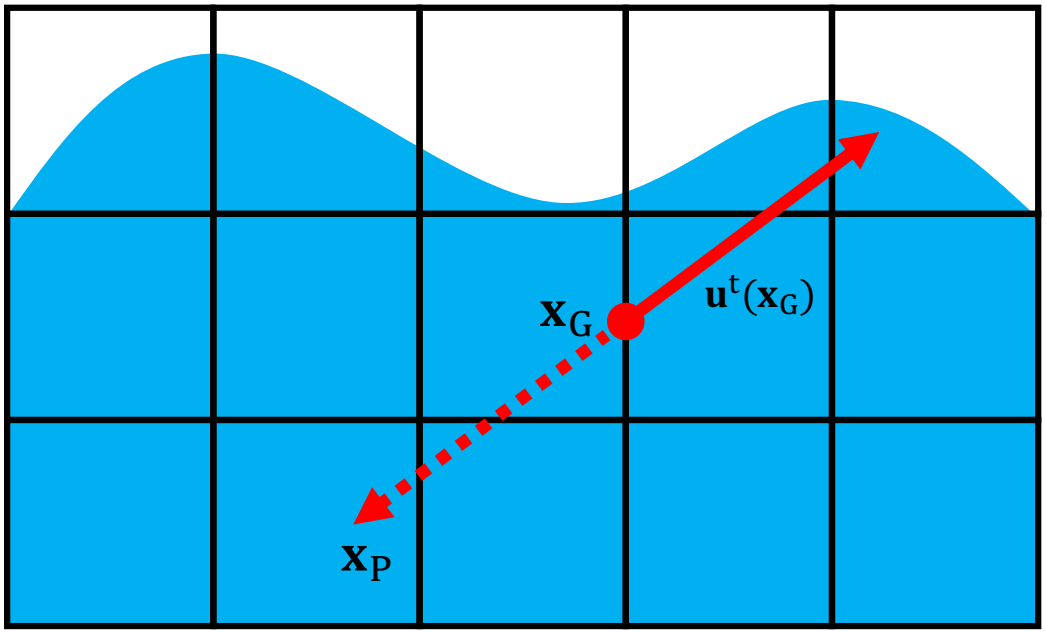
This term cause lots of problems

# Advection: Semi-Lagrangian




- To update the velocity at position $\mathbf{x}_G$

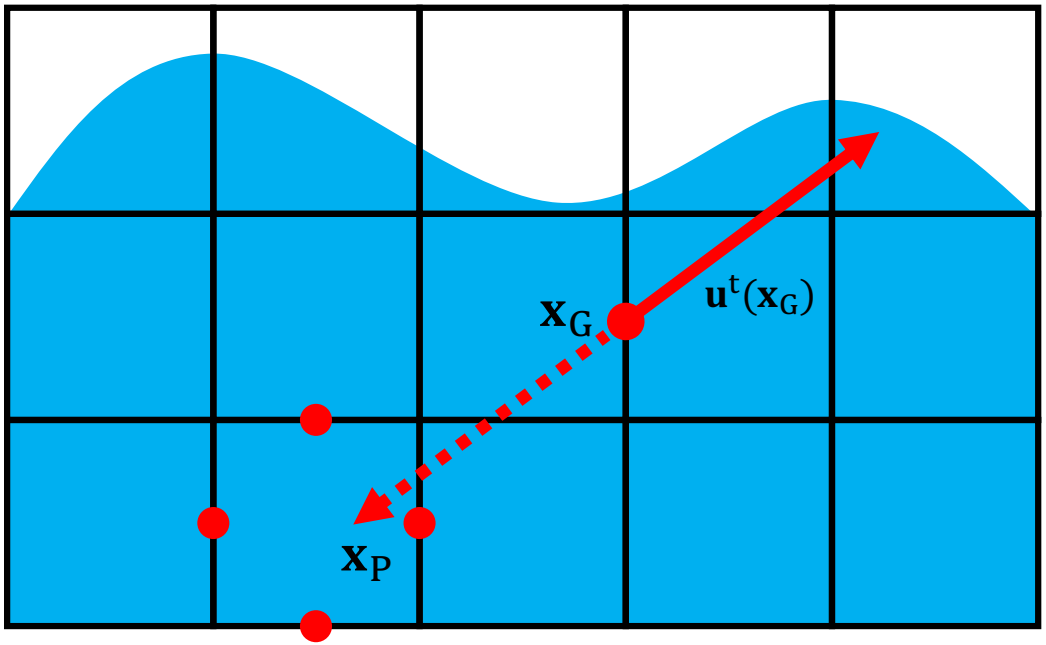
# Advection: Semi-Lagrangian




- Trace backward by the current velocity $\mathbf{u}^t(\mathbf{x}_G)$ as if a fluid particle is moving from $\mathbf{x}_P$ to $\mathbf{x}_G$

$$\mathbf{x}_P = \mathbf{x}_G - \Delta t \mathbf{u}^t(\mathbf{x}_G)$$

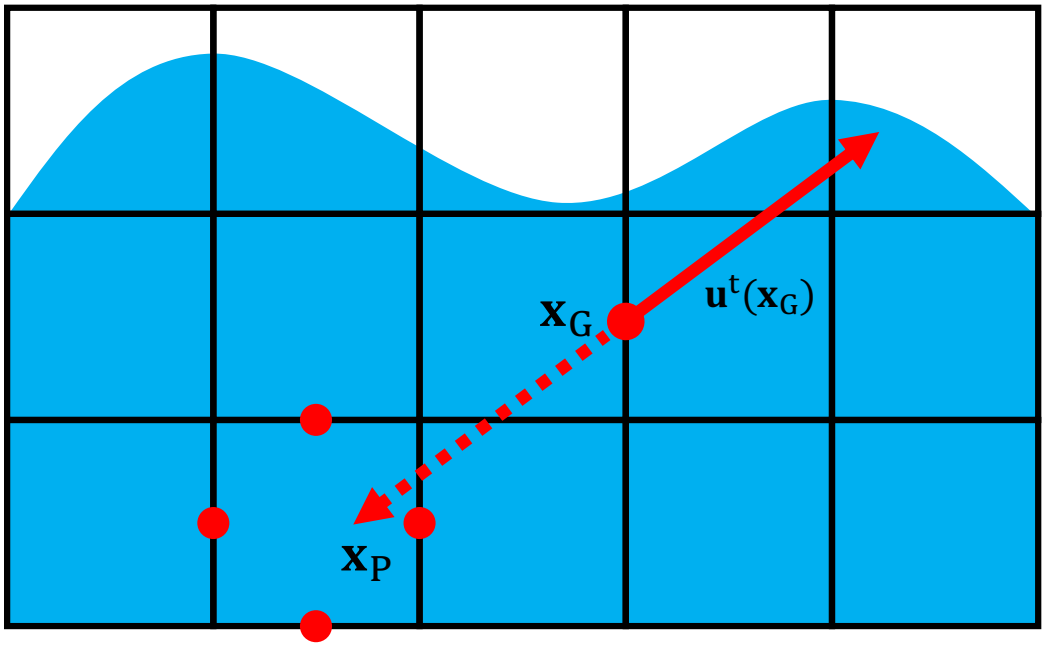
# Advection: Semi-Lagrangian




- Compute the velocity at $\mathbf{x}_p$ by interplating its neighbors' velocities, then assign it as the next velocity at $\mathbf{x}_G$

$$\mathbf{u}^{t+1}(\mathbf{x}_G) = \text{interpolate}(\mathbf{u}^t, \mathbf{x}_P)$$

# Advection: Semi-Lagrangian




$$\mathbf{x}_P = \mathbf{x}_G - \Delta t \mathbf{u}^t(\mathbf{x}_G)$$
$$\mathbf{u}^{t+1}(\mathbf{x}_G) = \text{interpolate}(\mathbf{u}^t, \mathbf{x}_P)$$

# Naïve vs. Semi-Lagrangian

- Naïve update is more like forward Euler integration whereas semi-Lagrangian is more like backward Euler integration

- Semi-Lagrangian with linear interpolation is ***unconditionally stable*** if simulation starts with valid states
  - Linear interpolation (convex combination) never generates larger values than the original values used as input

# Body Force

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \mathbf{g} - \frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u}$$

$$\mathbf{u}^{t+1}(\mathbf{x}) = \mathbf{u}^{t}(\mathbf{x}) + \Delta t \mathbf{g}$$

$$u_{i+1/2,j}^{t+1} = u_{i+1/2,j}^{t}$$

$$v_{i,j+1/2}^{t+1} = v_{i,j+1/2}^{t} + \Delta t(-9.81)$$

# Diffusion (Viscosity)

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \mathbf{g} - \frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u}$$

$$\mathbf{u}^{t+1}(\mathbf{x}) \leftarrow \mathbf{u}^{t}(\mathbf{x}) + \Delta t \nu \nabla^2 \mathbf{u}^{t}(\mathbf{x})$$

$$\nabla^2 \mathbf{u} = \frac{\mathbf{u}_{\text{right}} + \mathbf{u}_{\text{left}} + \mathbf{u}_{\text{top}} + \mathbf{u}_{\text{bottom}} - 4\mathbf{u}}{h^2}$$

- For water or gas simulation, this term is often ignored for two reasons:
  - They are not sticky enough
  - Artificial damping can be imposed by default during numerical integration when performing the advection step

# Projection (Pressure Solve)

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \mathbf{g} - \frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u}$$

$$\nabla \cdot \mathbf{u} = 0$$

- At this point, the pressure at the current timestep is not computed yet
- We want our velocity at the next timestep to be ***divergence-free (no compression)***, this should be compatible with the current pressure which we will computed as well

# Projection (Pressure Solve)

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \mathbf{g} - \frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u}$$

$$u_{i+1/2,j}^{t+1} = u_{i+1/2,j}^{t} - \Delta t \frac{1}{\rho} \frac{p_{i+1,j}^{t} - p_{i,j}^{t}}{h}$$

$$v_{i,j+1/2}^{t+1} = v_{i,j+1/2}^{t} - \Delta t \frac{1}{\rho} \frac{p_{i,j+1}^{t} - p_{i,j}^{t}}{h}$$

$$\nabla \cdot \mathbf{u} = 0$$

$$\nabla \cdot \mathbf{u}_{i,j}^{t+1} = \frac{u_{i+1/2,j}^{t+1} - u_{i-1/2,j}^{t+1}}{h} + \frac{v_{i,j+1/2}^{t+1} - v_{i,j-1/2}^{t+1}}{h} = 0$$

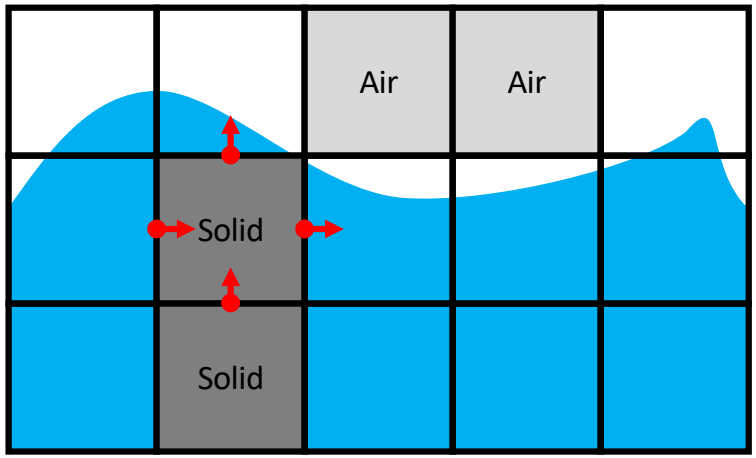
# Projection (Pressure Solve)

$$\frac{u_{i+1/2,j}^{t+1} - u_{i-1/2,j}^{t+1}}{h} + \frac{v_{i,j+1/2}^{t+1} - v_{i,j-1/2}^{t+1}}{h} = 0$$

$$\frac{1}{h}\left[\left(u_{i+1/2,j}^{t} - \Delta t \frac{1}{\rho}\frac{p_{i+1,j}^{t} - p_{i,j}^{t}}{h}\right) - \left(u_{i-1/2,j}^{t} - \Delta t \frac{1}{\rho}\frac{p_{i,j}^{t} - p_{i-1,j}^{t}}{h}\right)\right.$$
$$\left. + \left(v_{i,j+1/2}^{t} - \Delta t \frac{1}{\rho}\frac{p_{i,j+1}^{t} - p_{i,j}^{t}}{h}\right) - \left(v_{i,j-1/2}^{t} - \Delta t \frac{1}{\rho}\frac{p_{i,j}^{t} - p_{i,j-1}^{t}}{h}\right)\right] = 0$$

$$\frac{\Delta t}{\rho}\left(\frac{4p_{i,j}^{t} - p_{i+1,j}^{t} - p_{i,j+1}^{t} - p_{i-1,j}^{t} - p_{i,j-1}^{t}}{h^2}\right) = -\left(\frac{u_{i+1/2,j}^{t} - u_{i-1/2,j}^{t}}{h} + \frac{v_{i,j+1/2}^{t} - v_{i,j-1/2}^{t}}{h}\right)$$

# Boundary Conditions: Solid




- Fixed solid cells: $\mathbf{u}_{\text{solid}} = \mathbf{0}$
- Moving solid cells: $\mathbf{u}_{\text{solid}} \cdot \mathbf{n} = \mathbf{u}^{t+1} \cdot \mathbf{n}$

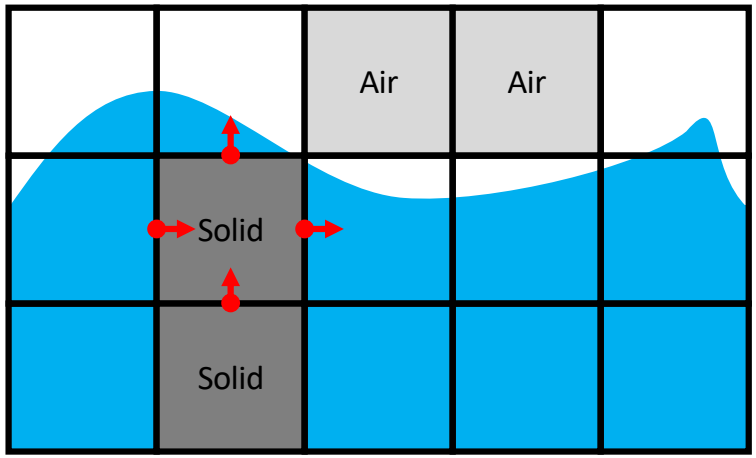
$$u_{\text{solid}} = u^t_{i+1/2,j} - \Delta t \frac{1}{\rho} \frac{p^t_{i+1,j} - p^t_{i,j}}{h}$$

$$v_{\text{solid}} = v^t_{i,j+1/2} - \Delta t \frac{1}{\rho} \frac{p^t_{i,j+1} - p^t_{i,j}}{h}$$

$$p^t_{i+1,j} - p^t_{i,j} = \frac{\rho h}{\Delta t}\left(u^t_{i+1/2,j} - u_{\text{solid}}\right)$$

$$p^t_{i,j+1} - p^t_{i,j} = \frac{\rho h}{\Delta t}\left(v^t_{i,j+1/2} - v_{\text{solid}}\right)$$

# Boundary Conditions: Air




- For air cells, simply set $p_{\mathrm{i,j}} = 0$

# Projection: Summary

- For each step of projection
  - Setup the linear system to find out the pressure field
    - When setting up the system, equation for each cell should be setup differently depending on its surrounding conditions; fluid, empty, or solid wall
  - Solve the linear system
    - The linear system should be ***sparse*** because each cell will be affected by its neighbor cells only
  - Update the velocity field using the computed pressures by using the known velocity-pressure equation

# Smoke

- To model the most important effects of smoke, we need two extra fluid variables: the ***temperature*** $T$ of the air and the ***concentration*** $s$ of smoke particles (the thing we actually see)

- The temperature and the concentration are typically stored in the grid cell centers (the same as pressure)

- In reality, temperature changes cause the air to slightly expand or contract, causing a change in density. However, these variations are small, so we will use what is called ***Boussinesq*** approximation which ignores density difference

# Smoke

- We will replace the gravitational force with a buoyancy force $f_{buoy}$ and assume buoyancy just depends linearly on the temperature and smoke concentration

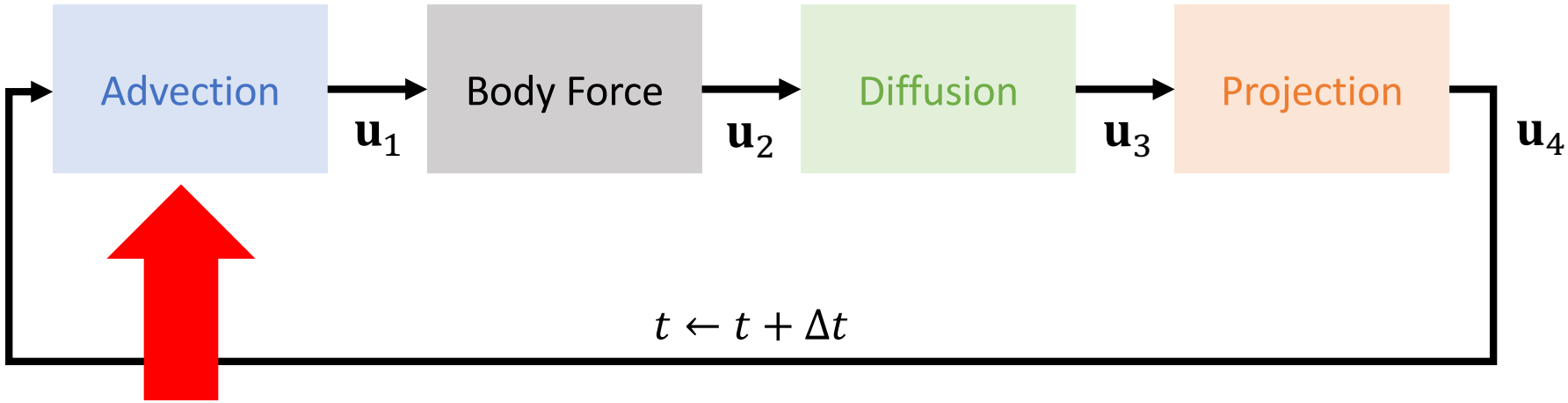
$$f_{buoy} = (0, -\alpha s + \beta(T - T_{amb}), 0)$$

where $\alpha$ and $\beta$ are nonnegative user-provided parameters

- The equation simply means that
  - When the temperature is high, smoke will move upward
  - If smoke particles are concentrated too much in the same cell, it will move down

# Algorithm (Splitting)

$$\underbrace{\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}}_{(1)} = \underbrace{f_{buoy}}_{(2)} \underbrace{- \frac{1}{\rho} \nabla p}_{(4)} + \underbrace{\nu \nabla^2 \mathbf{u}}_{(3)}$$




In addition to the velocity, the temperature and concentration will be advected as well $\frac{dT}{dt} = 0$ $\frac{ds}{dt} = 0$

# Numerical Dissipation (Energy Loss)

- Unless we run simulation with a very small timestep with a fine grid, simulation results will look overly smooth due to numerical dissipation (energy loss)

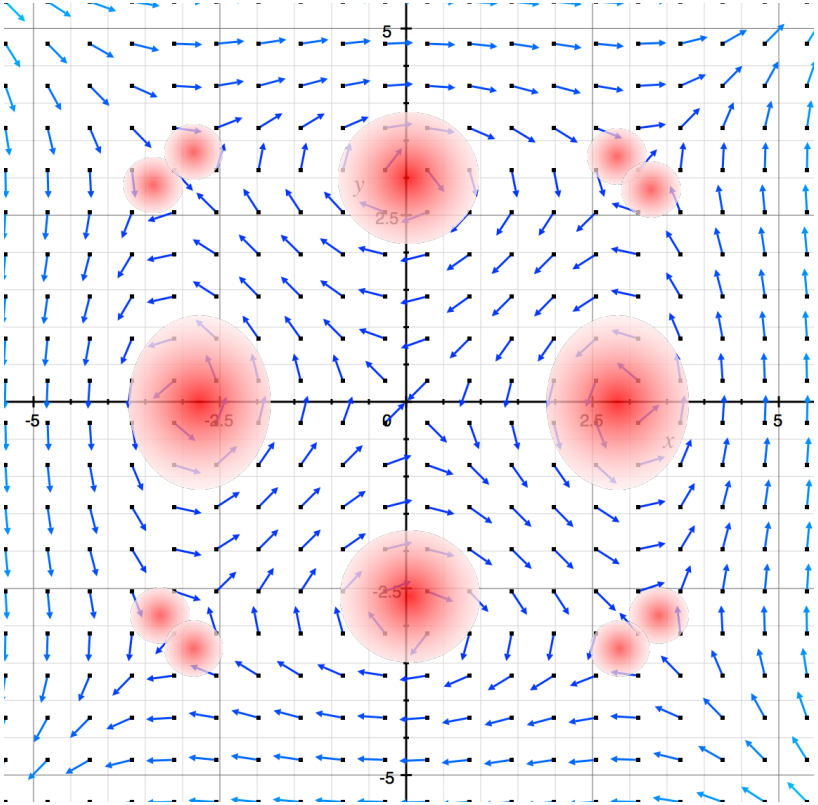
- The problem is that the interesting features such as small vortices, sharp devides between smoky and clean air simply vanish far faster than desired

# Vorticity Confinement

- To counter the problem of vortices artifically disappearing too fast, we can add a ***voriticy confinement*** force which boosts the strength of vortices in the flow, keeping it lively and interesting for much longer time

- Simply speaking, the idea is to add extra artificial engery to compensate energy loss, which helps visual enhancement

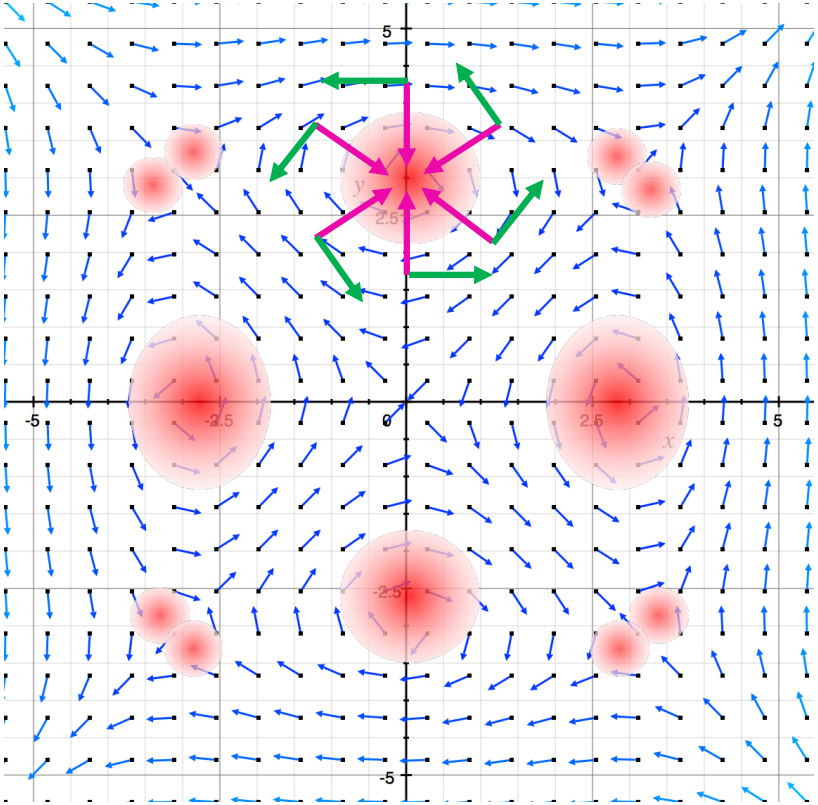
# Vorticity Confinement




- Given the current velocity field, we first measure ***vorticity***:

$$|\boldsymbol{\omega}| = |\nabla \times \mathbf{u}|$$

# Vorticity Confinement




- A vortex, loosely speaking, is a peak in the vorticity field, a place that's spinning faster than all the fluid nearby. We will construct unit vector $N$ that points to these peaks

$$N = \frac{\nabla|\boldsymbol{\omega}|}{\left|\nabla|\boldsymbol{\omega}|\right|}$$

$$f_{conf} = \epsilon h(N \times \boldsymbol{\omega})$$

$h$: grid size
$\epsilon$: tunable parameter

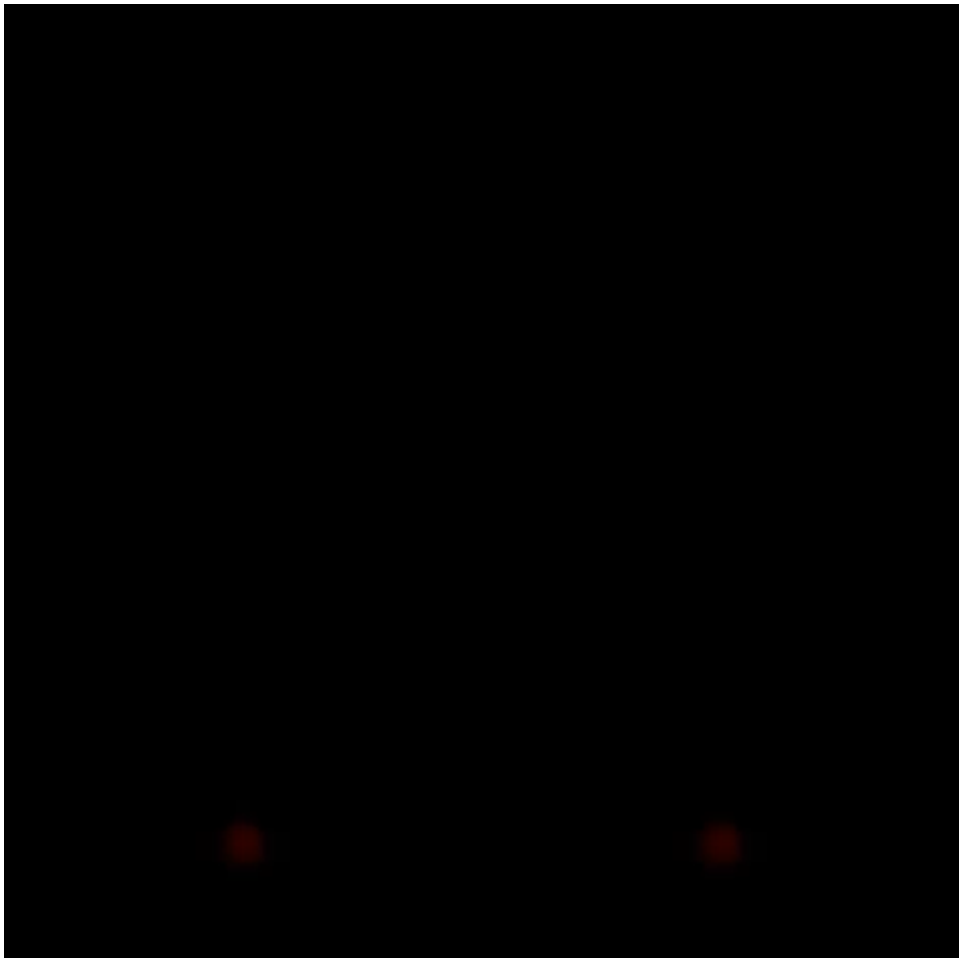
# Vorticity Confinement: Implementation

$$N_{i,j,k} = \frac{\nabla|\boldsymbol{\omega}|_{i,j,k}}{\left|\nabla|\boldsymbol{\omega}|_{i,j,k}\right| + 10^{-20}}$$

$$\nabla|\boldsymbol{\omega}|_{i,j,k} = \begin{pmatrix} \frac{|w|_{i+1,j,k} - |w|_{i-1,j,k}}{2h}, \\ \frac{|w|_{i,j+1,k} - |w|_{i,j-1,k}}{2h}, \\ \frac{|w|_{i,j,k+1} - |w|_{i,j,k-1}}{2h} \end{pmatrix} \quad \boldsymbol{\omega}_{i,j,k} = \begin{pmatrix} \frac{w_{i,j+1,k} - w_{i,j-1,k}}{2h} - \frac{w_{i,j,k+1} - w_{i,j,k-1}}{2h}, \\ \frac{u_{i,j,k+1} - u_{i,j,k-1}}{2h} - \frac{w_{i+1,j,k} - w_{i-1,j,k}}{2h}, \\ \frac{w_{i+1,j,k} - w_{i-1,j,k}}{2h} - \frac{w_{i,j+1,k} - w_{i,j-1,k}}{2h} \end{pmatrix}$$

We can simplifiy this to two dimensions by thinking of the two-dimensional velocity field as $(u, v, 0)$, which gives a vorticity field $\boldsymbol{\omega} = (0,0, w)$

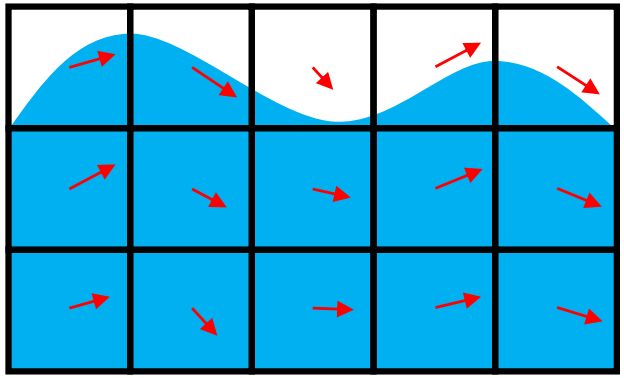
# Vorticity Confinement



Vorticity confinement (right) fire simulation - YouTube

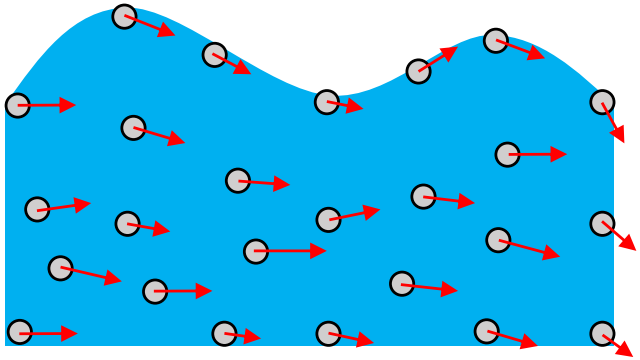
# Eulerian vs. Lagrangian

- Eulerian viewpoint



- We measure fluid quantities (e.g. velocity, pressure) in fixed locations

- Lagrangian viewpoint



- Particles represents fluid, where we measure fluid quantities each particle has

# NS Equation: Particle-based Simulation

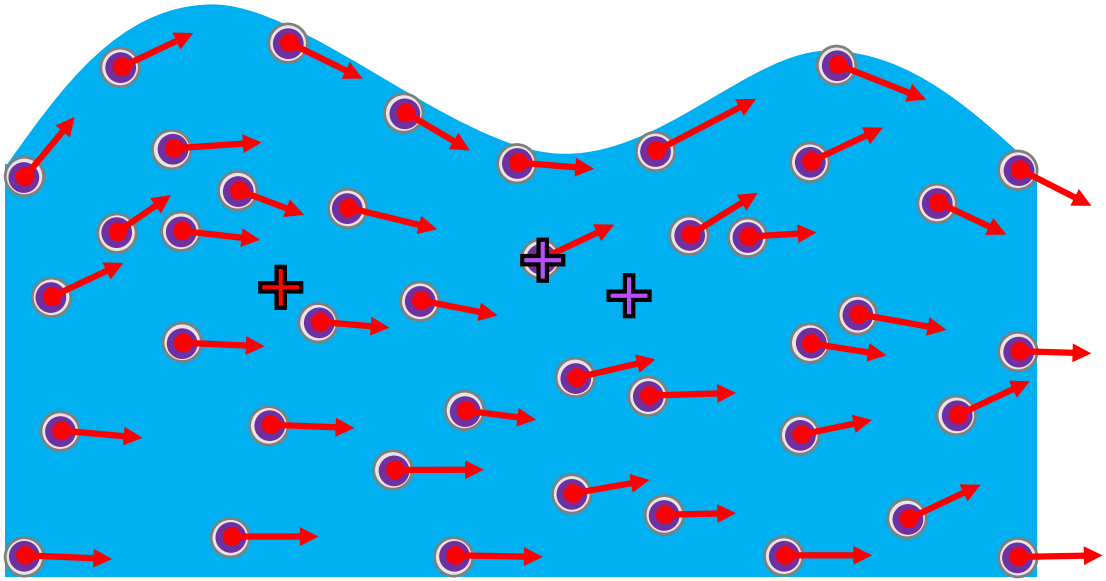
$$\frac{d\mathbf{u}}{dt} = \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \mathbf{g} - \frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u}$$

$$\nabla \cdot \mathbf{u} = 0$$

$$\frac{d\mathbf{u}}{dt} = \mathbf{g} - \frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u}$$

- PDE becomes ODE (assuming the viscosity is ignored)
- Many particle-based simulation does not enforce incompressibility

# Evaluation at an Arbitary Position: Particle-based Simulation



- Each particle carries over physical quantities (e.g. pressure, velocity), so the key question is ***how we can estimate physical quantities of the entire field from the particles***
  - Their positions could be irregular
  - The number of neighbors is dynamically changing and finding them requires additional computation

# Smoothed Particle Hydrodynamics (SPH)

- It was first proposed by [Gingold and Monaghan 1977] and [Lucy 1977] for modeling astrophysics (e.g. galaxy formation, star formation, stellar collisions)

- It is a meshfree (grid-free) Lagarangian method

# Smoothed Particle Hydrodynamics (SPH)

$$q(\mathbf{x}) = \int q(\mathbf{x}')W(|\mathbf{x}-\mathbf{x}'|,h)d\mathbf{x}'$$

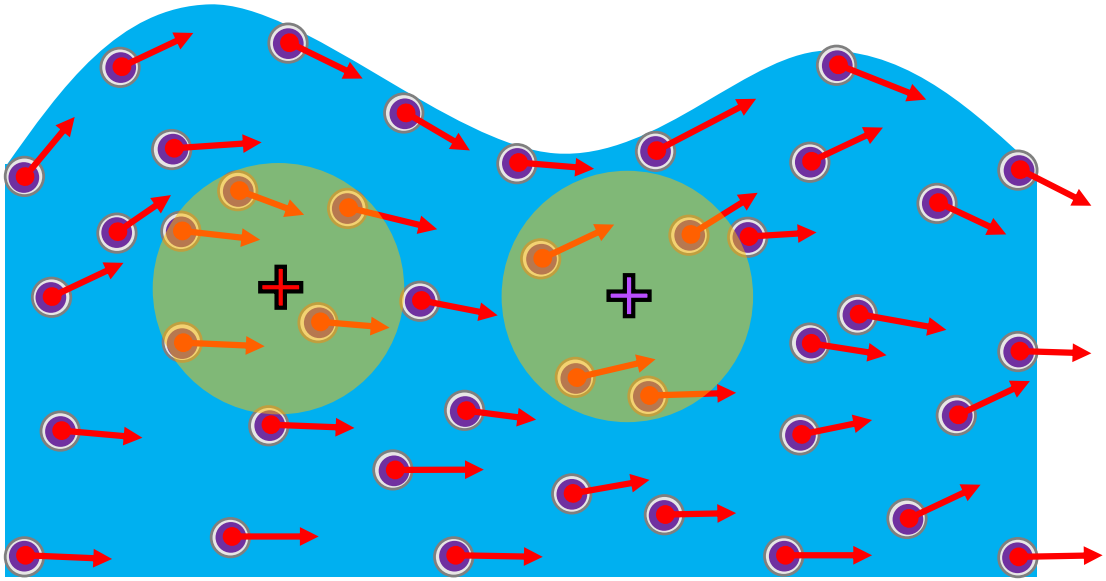
$$= \sum_{j\in N} q(\mathbf{x}_j)W(|\mathbf{x}-\mathbf{x}'|,h)dV_j$$

$$= \sum_{j\in N} \frac{\mathrm{m}_j}{\rho_j} q(\mathbf{x}_j)W(|\mathbf{x}-\mathbf{x}'|,h)$$

- Desired properties of kernel function
  - Compact property

$$1 = \int W(|\mathbf{x}-\mathbf{x}'|,h)d\mathbf{x}'$$

# Evaluation at an Arbitary Position: SPH



$$q(\mathbf{x}_i) = \sum_{j \in N} \frac{\mathrm{m}_j}{\rho_j} q(\mathbf{x}_j) W(|\mathbf{x} - \mathbf{x}'|, h)$$

$$f(\textcolor{red}{\boldsymbol{+}}) = Interp(\bullet_{N1}, \bullet_{N2}, \bullet_{N3}, \bullet_{N4})$$

$$f(\textcolor{red}{\boldsymbol{+}}) = Interp(\bullet\!\!\nearrow_{N1}, \bullet\!\!\nearrow_{N2}, \bullet\!\!\nearrow_{N3}, \bullet\!\!\nearrow_{N4}, \bullet\!\!\nearrow_{N5})$$

# Smoothed Particle Hydrodynamics (SPH)

- A popular choise is the kernel called *poly6*

$$W(|\mathbf{x}-\mathbf{x}'|,h) \equiv \frac{315}{64\pi h^9}(h^2-|\mathbf{x}-\mathbf{x}'|^2)^3$$

$$\nabla W(|\mathbf{x}-\mathbf{x}'|,h) \equiv \frac{-45}{\pi h^6}(h-|\mathbf{x}-\mathbf{x}'|)^2\frac{\mathbf{x}-\mathbf{x}'}{|\mathbf{x}-\mathbf{x}'|}$$

$$\nabla^2 W(|\mathbf{x}-\mathbf{x}'|,h) \equiv \frac{45}{\pi h^6}(h-|\mathbf{x}-\mathbf{x}'|)$$

  - $W = 0$ at distance $h$
  - It satisfies the compact property $1 = \int W(|\mathbf{x}-\mathbf{x}'|,h)d\mathbf{x}'$

# Smoothed Particle Hydrodynamics (SPH)

$$\frac{d\mathbf{u}}{dt} = \mathbf{g} - \frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u}$$

$$q(\mathbf{x}) = \sum_{j \in N} \frac{\mathrm{m}_j}{\rho_j} q(\mathbf{x}_j) W(|\mathbf{x} - \mathbf{x}_j|, h) \qquad \nabla q(\mathbf{x}) = \sum_{j \in N} \frac{\mathrm{m}_j}{\rho_j} q(\mathbf{x}_j) \nabla W(|\mathbf{x} - \mathbf{x}_j|, h)$$

$$\rho_i \approx \sum_{j \in N} \mathrm{m}_j W(|\mathbf{x} - \mathbf{x}_j|, h)$$

$$p_i \approx k(\rho_i - \rho_0)$$

$$\frac{\nabla p_i}{\rho_i} \approx \frac{1}{\rho_i} \sum_{j \in N} \frac{\mathrm{m}_j}{\rho_j} p_j \nabla W(|\mathbf{x} - \mathbf{x}_j|, h) \approx \frac{1}{\rho_i} \sum_{j \in N} \frac{\mathrm{m}_j}{\rho_j} \left(\frac{p_i + p_j}{2}\right) \nabla W(|\mathbf{x} - \mathbf{x}_j|, h)$$

$$\nu\nabla^2\mathbf{u}_i \approx \nu \sum_{j \in N} \frac{\mathrm{m}_j}{\rho_j} \mathbf{u}_j \nabla^2 W(|\mathbf{x} - \mathbf{x}_j|, h) \approx \nu \sum_{j \in N} \frac{\mathrm{m}_j}{\rho_j} (\mathbf{u}_j - \mathbf{u}_i) \nabla^2 W(|\mathbf{x} - \mathbf{x}_j|, h)$$

# Algorithm: SPH

**while** simulating:

**for** each particle $i$:

Compute density $\rho_i$

Compute pressure $p_i = k(\rho - \rho_0)$

Compute pressure gradient $\frac{\nabla p_i}{\rho_i}$

Compute viscosity term $\nu \nabla^2 \mathbf{u}_i$

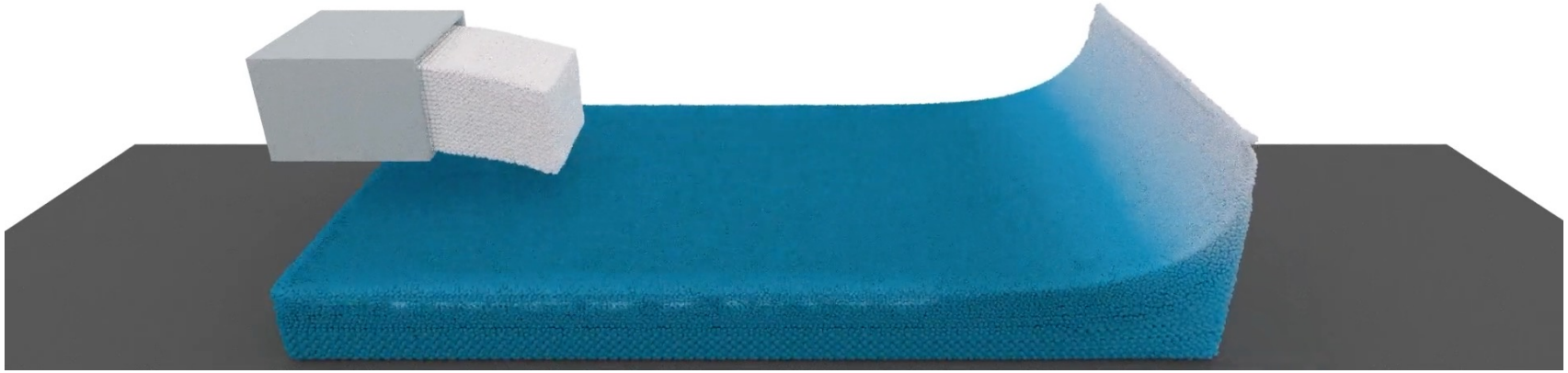
Add body force (e.g. gravity)

Compute acceleration $\frac{d\mathbf{u}}{dt}$

Numerically integrate its velocity and position

- Computational bottleneck is to compute pair-wise distances between all the particles
- The amount of computation can be reduced by partitioning space into local regions

# Video: SPH



(1) openMaelstrom: an openSource SPH simulation and rendering framework - YouTube